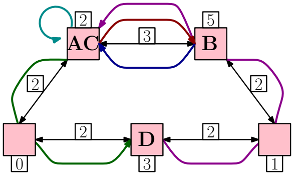
Virtual Network Embedding Approximations: Leveraging Randomized Rounding



IFIP Networking 2018

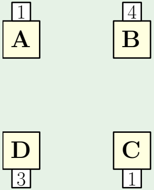
Matthias Rost
Technische Universität Berlin, Internet Network Architectures

Stefan Schmid
Universität Wien, Communication Technologies

Introduction: Virtual Network Embeddings

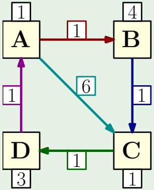
'Classic' Cloud Computing

- Only number and 'size' of virtual machines is given
- No guarantee on network performance



Goal: Virtual Networks (since ≈ 2006)

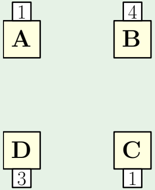
- Additionally: communication requirements given
- Network performance will be guaranteed



Introduction: Virtual Network Embeddings

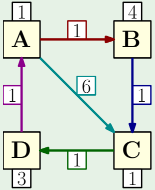
'Classic' Cloud Computing

- Only number and 'size' of virtual machines is given
- No guarantee on network performance



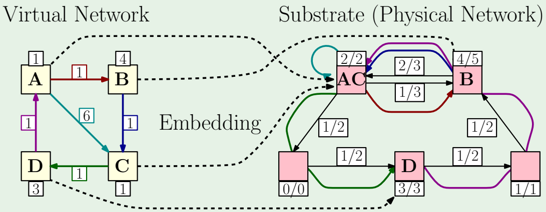
Goal: Virtual Networks (since ≈ 2006)

- Additionally: communication requirements given
- Network performance will be guaranteed



Embedding of Virtual Networks

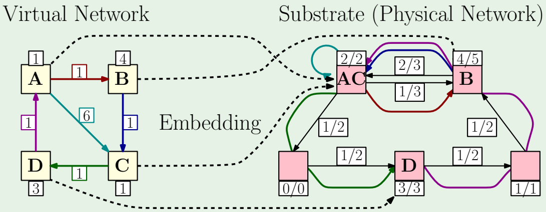
- Map virtual nodes to substrate nodes
- Map virtual edges to paths in the substrate
- Respecting mapping restrictions
- Respecting capacities



Introduction: Virtual Network Embeddings

Embedding of Virtual Networks

- Map virtual nodes to substrate nodes
- Map virtual edges to paths in the substrate
- Respecting mapping restrictions
- Respecting capacities



Virtual Network Embedding Problem (VNEP) ≈ 2006

Online: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

Offline: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

Introduction: Virtual Network Embeddings

Virtual Network Embedding Problem (VNEP) ≈ 2006

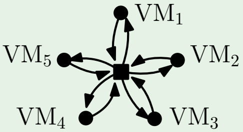
Online: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

Offline: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

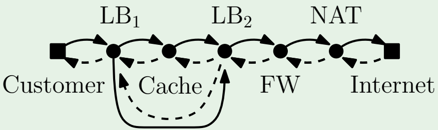
Importance of the Virtual Network Embedding Problem

- Studied extensively over the last decade ($>$ 100 publications)
- 'Parent' to Virtual Cluster Embeddings (≈ 2011) and Service Chain Embeddings (≈ 2013)

Virtual Cluster



Service Chain



Introduction: Virtual Network Embeddings

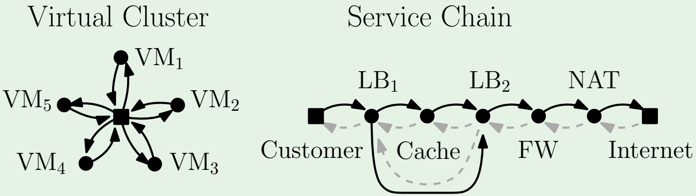
Virtual Network Embedding Problem (VNEP) ≈ 2006

Online: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

Offline: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

Importance of the Virtual Network Embedding Problem

- Studied extensively over the last decade ($>$ 100 publications)
- 'Parent' to Virtual Cluster Embeddings (≈ 2011) and Service Chain Embeddings (≈ 2013)



cactus graphs: cycles intersect in at most one node

Introduction: Virtual Network Embeddings

Virtual Network Embedding Problem (VNEP) $\approx$ 2006

Online: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

Offline: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

Algorithmic Approaches to the VNEP

Heuristics

- no quality guarantee
- polynomial-time
- respects all constraints
- very intensively studied

Approximation Algorithms

- quality guarantee
- polynomial-time
- cannot respect all constraints[1]
- not studied for general request graphs

Exact Algorithms

- near-optimal solutions
- exponential-time
- respects all constraints
- intensively studied

Introduction: Virtual Network Embeddings

Virtual Network Embedding Problem (VNEP) $\approx$ 2006

Online: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

Offline: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

Algorithmic Approaches to the VNEP

Heuristics

- no quality guarantee
- polynomial-time
- respects all constraints
- very intensively studied

Approximation Algorithms

- **quality guarantee**
- **polynomial-time**
- **cannot respect all constraints**[1]
- **not studied** for general request graphs

Exact Algorithms

- near-optimal solutions
- exponential-time
- respects all constraints
- intensively studied

Introduction: Virtual Network Embeddings

Virtual Network Embedding Problem (VNEP) ≈ 2006

Online: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

Offline: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

Algorithmic Approaches to the VNEP

Heuristics

- no quality guarantee
- polynomial-time
- respects all constraints
- very intensively studied

Approximation Algorithms

- **quality guarantee**
- **polynomial-time**
- **cannot respect all constraints**[1]
- **not studied** for general request graphs

Exact Algorithms

- near-optimal solutions
- exponential-time
- respects all constraints
- intensively studied

Introduction: Virtual Network Embeddings

Virtual Network Embedding Problem (VNEP) $\approx$ 2006

Online: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

Offline: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

Algorithmic Approaches to the VNEP

Heuristics

- no quality guarantee
- polynomial-time
- respects all constraints
- very intensively studied

Approximation Algorithms

- **quality guarantee**
- **polynomial-time**
- **cannot respect all constraints**[1]
- **not studied** for general request graphs

Exact Algorithms

- near-optimal solutions
- exponential-time
- respects all constraints
- intensively studied

[1] Matthias Rost and Stefan Schmid. "Charting the Complexity Landscape of Virtual Network Embeddings". In: *Proc. IFIP Networking*. 2018

Introduction: Virtual Network Embeddings

Virtual Network Embedding Problem (VNEP) $\approx$ 2006

Online: Find an optimal feasible embedding for a single request (e.g. minimizing resource cost).

Offline: Find feasible embeddings for an optimal (sub)set of requests (e.g. maximizing achieved profit).

Algorithmic Approaches to the VNEP

Heuristics

- no quality guarantee
- polynomial-time
- respects all constraints
- very intensively studied

Approximation Algorithms

- **quality guarantee**
- **polynomial-time**
- **cannot respect all constraints**[1]
- **not studied** for general request graphs

Exact Algorithms

- near-optimal solutions
- exponential-time
- respects all constraints
- intensively studied

---

[1] Matthias Rost and Stefan Schmid. "Charting the Complexity Landscape of Virtual Network Embeddings". In: *Proc. IFIP Networking*. 2018

Contributions

Heuristics

- no quality guarantee
- polynomial-time
- respects all constraints
- very intensively studied

Approximation Algorithms

- **quality guarantee**
- **polynomial-time**
- **cannot respect all constraints**[1]
- **not studied** for general request graphs

Exact Algorithms

- near-optimal solutions
- exponential-time
- respects all constraints
- intensively studied

Contributions of our paper

1. **First approximation algorithm for the offline VNEP for maximizing the profit**[a]**.**
2. **Derived heuristics and studied performance in extensive computational study.**

[a] For a limited class of request graphs: cactus graphs

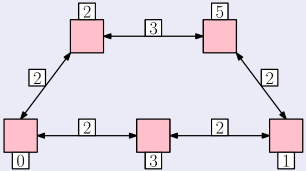
[1] Matthias Rost and Stefan Schmid. "Charting the Complexity Landscape of Virtual Network Embeddings". In: *Proc. IFIP Networking*. 2018

Formal Problem Statement & Integer Program

Formal Problem Statement & Integer Program

Substrate Network

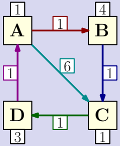
- Capacitated graph $G_S = (V_S, E_S)$



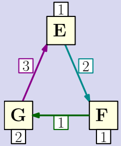
For each request $r \in \mathcal{R}$ ...

- Capacitated graph $G_r = (V_r, E_r)$
- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

Request 1: G_1



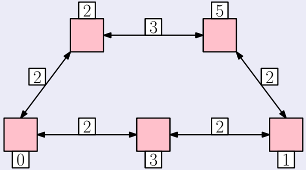
Request 2: G_2



Formal Problem Statement & Integer Program

Substrate Network

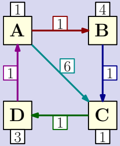
- Capacitated graph $G_S = (V_S, E_S)$



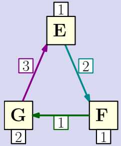
For each request $r \in \mathcal{R}$ ...

- Capacitated graph $G_r = (V_r, E_r)$
- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

Request 1: G_1



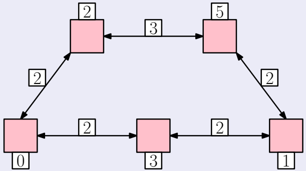
Request 2: G_2



Formal Problem Statement & Integer Program

Substrate Network

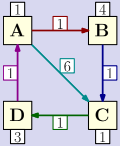
- Capacitated graph $G_S = (V_S, E_S)$



For each request $r \in \mathcal{R}$ ...

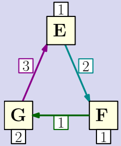
- Capacitated graph $G_r = (V_r, E_r)$
- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

Request 1: G_1



100$

Request 2: G_2

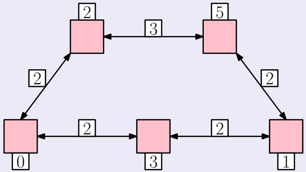


50$

Formal Problem Statement & Integer Program

Substrate Network

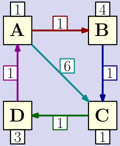
- Capacitated graph $G_S = (V_S, E_S)$



For each request $r \in \mathcal{R}$ ...

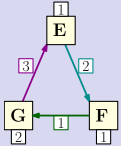
- Capacitated graph $G_r = (V_r, E_r)$
- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

Request 1: G_1



100$

Request 2: G_2



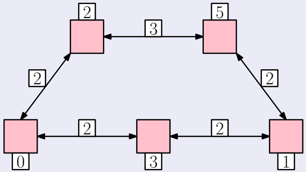
50$

Valid mappings: **single virtual element mappings** do not violate resource or mapping restrictions.

Formal Problem Statement & Integer Program

Substrate Network

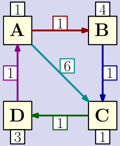
- Capacitated graph $G_S = (V_S, E_S)$



For each request $r \in \mathcal{R}$ ...

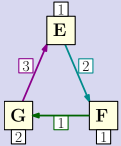
- Capacitated graph $G_r = (V_r, E_r)$
- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

Request 1: G_1



100$

Request 2: G_2

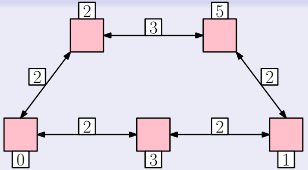


50$

Valid mappings: **single virtual element mappings** do not violate resource or mapping restrictions.

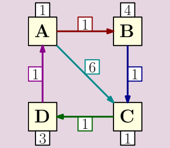
Formal Problem Statement & Integer Program

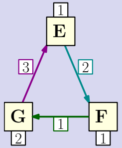
Substrate Network



For each request $r \in \mathcal{R}$ ...

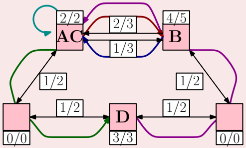
- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

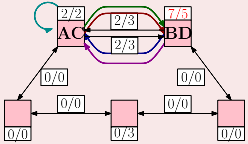


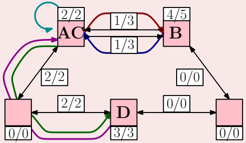


Valid mappings: **single virtual element mappings** do not violate resource or mapping restrictions.

Valid mappings for request 1: $\mathcal{M}_1 = \{m_2^1, m_2^2, m_3^3, \ldots\}$



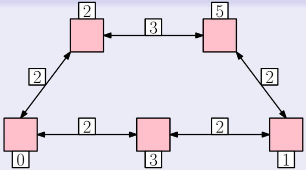




...

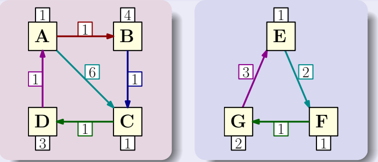
Formal Problem Statement & Integer Program

Substrate Network



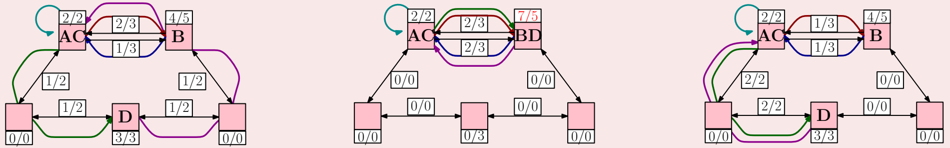
For each request $r \in \mathcal{R}$ ...

- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$



Valid mappings: **single virtual element mappings** do not violate resource or mapping restrictions.

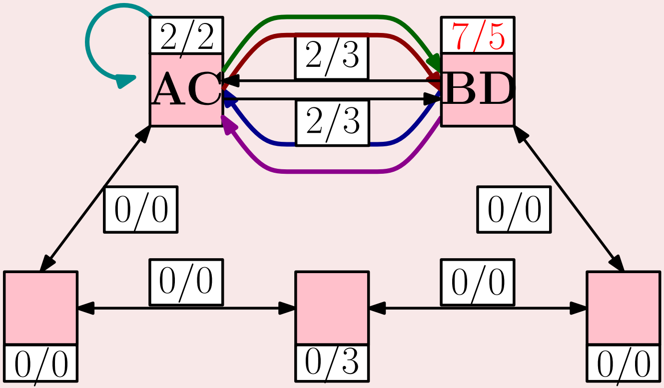
Valid mappings for request 1: $\mathcal{M}_1 = \{m_2^1, m_2^2, m_3^3, \ldots\}$



. . .

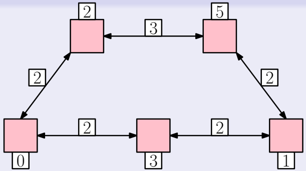
Formal Problem Statement & Integer Program

est 1: $\mathcal{M}_1 = \{m_2^1, m_2^2, m_3^3, \ldots\}$



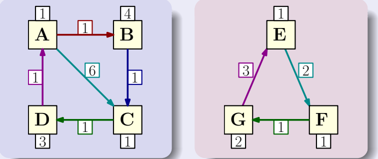
Formal Problem Statement & Integer Program

Substrate Network



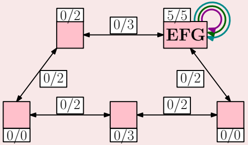
For each request $r \in \mathcal{R}$ ...

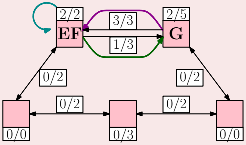
- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$

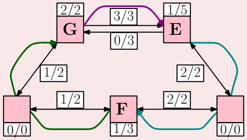


Valid mappings: **single virtual element mappings** do not violate resource or mapping restrictions.

Valid mappings for request 2: $\mathcal{M}_2 = \{m_2^1, m_2^2, m_3^3, \ldots\}$



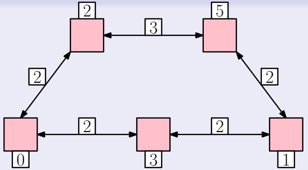




...

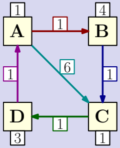
Formal Problem Statement & Integer Program

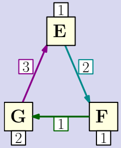
Substrate Network



For each request $r \in \mathcal{R}$ ...

- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$





Virtual Network Embedding Problem as Integer Program

- Is k-th mapping of request r chosen?

$$f_r^k \in \{0,1\} \qquad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \quad (1)$$

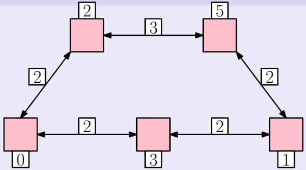
$$\sum_{m_r^k \in \mathcal{M}_r} f_r^k \leq 1 \qquad \forall r \in \mathcal{R} \quad (2)$$

$$\sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \qquad \forall x \in R_S \quad (3)$$

$$\max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k \quad (4)$$

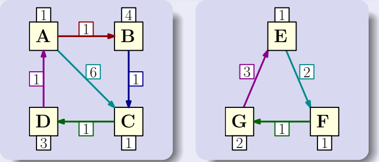
Formal Problem Statement & Integer Program

Substrate Network



For each request $r \in \mathcal{R}$ ...

- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$



Virtual Network Embedding Problem as Integer Program

- Is k-th mapping of request r chosen?
- Select at most one mapping:

$$f_r^k \in \{0,1\} \qquad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \tag{1}$$

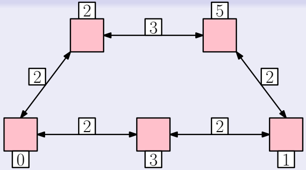
$$\sum_{m_r^k \in \mathcal{M}_r} f_r^k \leq 1 \qquad \forall r \in \mathcal{R} \tag{2}$$

$$\sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \qquad \forall x \in R_S \tag{3}$$

$$\max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k \tag{4}$$

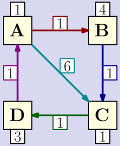
Formal Problem Statement & Integer Program

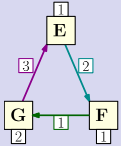
Substrate Network



For each request $r \in \mathcal{R}$ ...

- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$





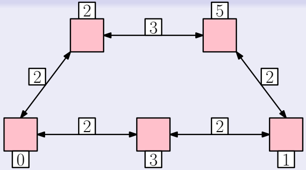
Virtual Network Embedding Problem as Integer Program

- Is k-th mapping of request r chosen? $$f_r^k \in \{0,1\} \qquad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \qquad (1)$$
- Select at most one mapping: $$\sum_{m_r^k \in \mathcal{M}_r} f_r^k \leq 1 \qquad \forall r \in \mathcal{R} \qquad (2)$$
- Enforce capacity for each resource x: $$\sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \qquad \forall x \in R_S \qquad (3)$$

$$\max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k \qquad (4)$$

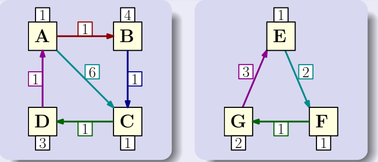
Formal Problem Statement & Integer Program

Substrate Network



For each request $r \in \mathcal{R}$ ...

- Mapping restrictions
- Profit $p_r > 0$
- Valid mappings $\mathcal{M}_r$



Virtual Network Embedding Problem as Integer Program

- Is k-th mapping of request r chosen? $$f_r^k \in \{0,1\} \quad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \quad (1)$$
- Select at most one mapping: $$\sum_{m_r^k \in \mathcal{M}_r} f_r^k \leq 1 \quad \forall r \in \mathcal{R} \quad (2)$$
- Enforce capacity for each resource x: $$\sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \quad \forall x \in R_S \quad (3)$$
- Maximize the profit: $$\max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k \quad (4)$$

Formal Problem Statement & Integer Program

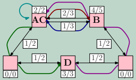
Virtual Network Embedding Problem as Integer Program

- Is k-th mapping of request r chosen? $\quad f_r^k \in \{0,1\} \quad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \quad (1)$
- Select at most one mapping: $\quad \sum_{m_r^k \in \mathcal{M}_r} f_r^k \leq 1 \quad \forall r \in \mathcal{R} \quad (2)$
- Enforce capacity for each resource x: $\quad \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \quad \forall x \in R_S \quad (3)$
- Maximize the profit: $\quad \max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k \quad (4)$

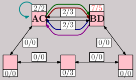
Example Solution to Integer Program: Profit 100$

Variables of request 1

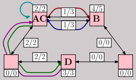
$f_1^1 = 1$



$f_1^2 = 0$



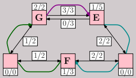
$f_1^3 = 0$



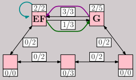
...

Variables of request 2

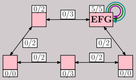
$f_2^1 = 0$



$f_2^2 = 0$



$f_2^3 = 0$



...

Formal Problem Statement & Integer Program

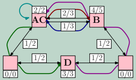
Virtual Network Embedding Problem as Integer Program

- Is k-th mapping of request r chosen? $$f_r^k \in \{0,1\} \quad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \quad (1)$$
- Select at most one mapping: $$\sum_{m_r^k \in \mathcal{M}_r} f_r^k \leq 1 \quad \forall r \in \mathcal{R} \quad (2)$$
- Enforce capacity for each resource x: $$\sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \quad \forall x \in R_S \quad (3)$$
- Maximize the profit: $$\max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k \quad (4)$$

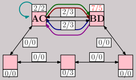
Example Solution to Integer Program: Profit 100$

Variables of request 1

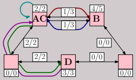
$f_1^1 = 1$



$f_1^2 = 0$



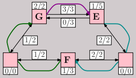
$f_1^3 = 0$



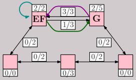
. . .

Variables of request 2

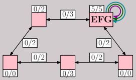
$f_2^1 = 0$



$f_2^2 = 0$



$f_2^3 = 0$



. . .

Approximation Framework: Randomized Rounding[2]

[2] P Raghavan and C D Thompson. "Provably Good Routing in Graphs: Regular Arrays". In: *Proc. 17th ACM STOC*. 1985, pp. 79–87.

Approximation Framework: Randomized Rounding

Assumption (for now):

Sets of valid mappings are of polynomial size and given.
$\Rightarrow$ LP Formulation can be solved in polynomial-time.

Virtual Network Embedding Problem as Linear Program

- Is k-th mapping of request r chosen? $$f_r^k \in [0,1] \qquad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \qquad (5)$$
- Select at most one mapping: $$\sum_{m_r^k \in \mathcal{M}_r} f_r^k \leq 1 \qquad \forall r \in \mathcal{R} \qquad (6)$$
- Enforce capacity for each resource x: $$\sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} A(m_r^k, x) \cdot f_r^k \leq c_S(x) \qquad \forall x \in R_S \qquad (7)$$
- Maximize the profit: $$\max \sum_{r \in \mathcal{R}} \sum_{m_r^k \in \mathcal{M}_r} p_r f_r^k \qquad (8)$$

Approximation Framework: Randomized Rounding

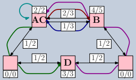
Virtual Network Embedding Problem as Linear Program

- Is k-th mapping of request r chosen? $\quad f_r^k \in [0,1] \quad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \quad (5)$
- ... $\quad \dots$

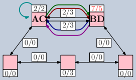
Example Solution to Linear Program: Profit 133$

Variables of request 1

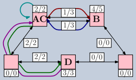
$f_1^1 = 0.5$



$f_1^2 = 0.3$



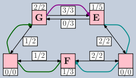
$f_1^3 = 0.2$



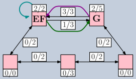
...

Variables of request 2

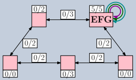
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Approximation Framework: Randomized Rounding

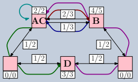
Virtual Network Embedding Problem as Linear Program

- Is k-th mapping of request r chosen? $\quad f_r^k \in [0,1] \qquad \forall r \in \mathcal{R}, m_r^k \in \mathcal{M}_r \quad (5)$
- … $\quad$ …

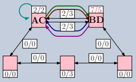
Example Solution to Linear Program: Profit 133$

Variables of request 1

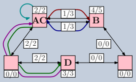
$f_1^1 = 0.5$



$f_1^2 = 0.3$



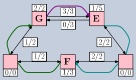
$f_1^3 = 0.2$



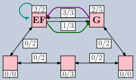
…

Variables of request 2

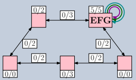
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



…

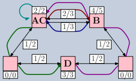
LP solution is convex combination valid mappings!

Let $\mathcal{D}_r = \{(f_r^k, m_r^k) | f_r^k > 0, m_r^k \in \mathcal{M}_r\}$ denote these optimal convex combinations for request r.

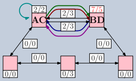
Example Solution to Linear Program: Profit 133$

Variables of request 1

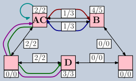
$f_1^1 = 0.5$



$f_1^2 = 0.3$



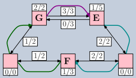
$f_1^3 = 0.2$



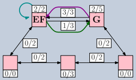
...

Variables of request 2

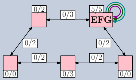
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

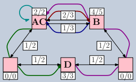
return *solution*

Approximation Framework: Randomized Rounding

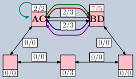
Example Solution to Linear Program: Profit 133$

Variables of request 1

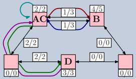
$f_1^1 = 0.5$



$f_1^2 = 0.3$



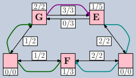
$f_1^3 = 0.2$



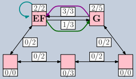
...

Variables of request 2

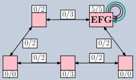
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

return *solution*

Rounding Outcomes

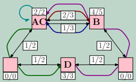
Iter.	Req. 1	Req. 2	Profit	max Load

Approximation Framework: Randomized Rounding

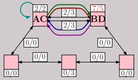
Example Solution to Linear Program: Profit 133$

Variables of request 1

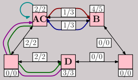
$f_1^1 = 0.5$



$f_1^2 = 0.3$



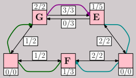
$f_1^3 = 0.2$



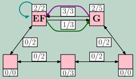
...

Variables of request 2

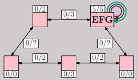
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

return *solution*

Rounding Outcomes

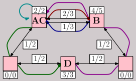
Iter.	Req. 1	Req. 2	Profit	max Load
1	m_1^1	m_2^2	150$	200%

Approximation Framework: Randomized Rounding

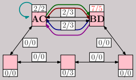
Example Solution to Linear Program: Profit 133$

Variables of request 1

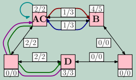
$f_1^1 = 0.5$



$f_1^2 = 0.3$



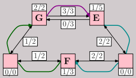
$f_1^3 = 0.2$



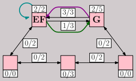
...

Variables of request 2

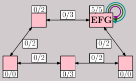
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

return *solution*

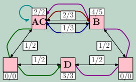
Rounding Outcomes

Iter.	Req. 1	Req. 2	Profit	max Load
1	m_1^1	m_2^2	150$	200%
2	m_1^3	$\emptyset$	100$	100%

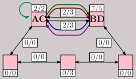
Example Solution to Linear Program: Profit 133$

Variables of request 1

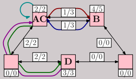
$f_1^1 = 0.5$



$f_1^2 = 0.3$



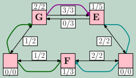
$f_1^3 = 0.2$



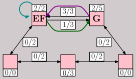
...

Variables of request 2

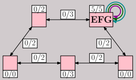
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

return *solution*

Rounding Outcomes

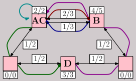
Iter.	Req. 1	Req. 2	Profit	max Load
1	m_1^1	m_2^2	150$	200%
2	m_1^3	$\emptyset$	100$	100%
3	m_1^1	m_2^1	150$	200%

Approximation Framework: Randomized Rounding

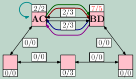
Example Solution to Linear Program: Profit 133$

Variables of request 1

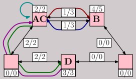
$f_1^1 = 0.5$



$f_1^2 = 0.3$



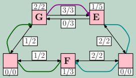
$f_1^3 = 0.2$



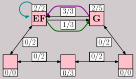
...

Variables of request 2

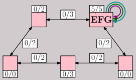
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

return *solution*

Rounding Outcomes

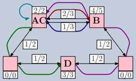
Iter.	Req. 1	Req. 2	Profit	max Load
1	m_1^1	m_2^2	150$	200%
2	m_1^3	$\emptyset$	100$	100%
3	m_1^1	m_2^1	150$	200%
4	m_1^2	m_2^1	150$	200%

Approximation Framework: Randomized Rounding

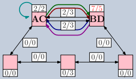
Example Solution to Linear Program: Profit 133$

Variables of request 1

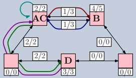
$f_1^1 = 0.5$



$f_1^2 = 0.3$



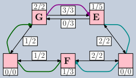
$f_1^3 = 0.2$



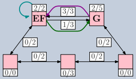
...

Variables of request 2

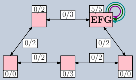
$f_2^1 = 0.5$



$f_2^2 = 0.16$



$f_2^3 = 0$



...

Idea: Treat weights as probabilities!

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$

foreach $r \in \mathcal{R}$ **do**

  choose m_r^k **with probability** f_r^k

end

return *solution*

Rounding Outcomes

Iter.	Req. 1	Req. 2	Profit	max Load
1	m_1^1	m_2^2	150$	200%
2	m_1^3	$\emptyset$	100$	100%
3	m_1^1	m_2^1	150$	200%
4	m_1^2	m_2^1	150$	200%
⋮	⋮	⋮	⋮	⋮

Approximation Algorithm for VNEP & Derived Heuristics

Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation

```
// perform preprocessing
```
compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r\in\mathcal{R}}$ from LP solution
do
  solution $\leftarrow$ RoundingProcedure($\{\mathcal{D}_r\}_{r\in\mathcal{R}}$)
while (solution *not* (α, β, γ)-approximate and rounding tries not exceeded)

Algorithm: RoundingProcedure

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r\in\mathcal{R}}$
foreach $r \in \mathcal{R}$ **do**
  choose m_r^k **with probability** f_r^k
end
return *solution*

Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation

```
// perform preprocessing
```

compute *optimal* LP solution

compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution

do

  solution $\leftarrow$ RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)

while (solution *not* (α, β, γ)-approximate and rounding tries not exceeded)

Main Theorem: First Approximation for the Virtual Network Embedding Problem

The Algorithm returns (α, β, γ)-approximate solutions for the VNEP[a] of at least an α fraction of the optimal profit, and allocations on nodes and edges within factors of β and γ of the original capacities, respectively, *with high probability*.

[a] restricted on cactus request graphs

Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation

```
// perform preprocessing
```
compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r\in\mathcal{R}}$ from LP solution
do
  solution $\leftarrow$ RoundingProcedure($\{\mathcal{D}_r\}_{r\in\mathcal{R}}$)
while (solution *not* (α,β,γ)-approximate and rounding tries not exceeded)

Definition of Parameters

$$\alpha = 1/3 \quad \text{(relative achieved profit)}$$

$$\beta = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(R^V_S) \cdot \log(|R^V_S|)}) \quad \text{(max node load)}$$

$$\gamma = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(E_S) \cdot \log(|E_S|)}) \quad \text{(max edge load)}$$

$$\varepsilon = \max_{r\in\mathcal{R}, x\in R_S} d_{\max}(r,x)/c_S(x) \leq 1 \quad \text{(max demand/capacity)}$$

$$\Delta(X) = \max_{x\in X} \sum_{r\in\mathcal{R}} (A_{\max}(r,x)/d_{\max}(r,x))^2 \quad \left(\begin{array}{c}\text{sum over } \mathcal{R} \text{ of squared} \\ \text{max (total / single) alloc}\end{array}\right)$$

Main Theorem: First Approximation for the Virtual Network Embedding Problem

The Algorithm returns (α,β,γ)-approximate solutions for the VNEP[a] of at least an α fraction of the optimal profit, and allocations on nodes and edges within factors of β and γ of the original capacities, respectively, *with high probability*.

[a] restricted on cactus request graphs

Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation

```
// perform preprocessing
```
compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution
do
  solution ← RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)
while (solution *not* (α, β, γ)-approximate and rounding tries not exceeded)

Definition of Parameters

$$\alpha = 1/3 \quad \text{(relative achieved profit)}$$

$$\beta = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(R_S^V) \cdot \log(|R_S^V|)}) \quad \text{(max node load)}$$

$$\gamma = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(E_S) \cdot \log(|E_S|)}) \quad \text{(max edge load)}$$

$$\varepsilon = \max_{r \in \mathcal{R}, x \in R_S} d_{\max}(r, x)/c_S(x) \leq 1 \quad \text{(max demand/capacity)}$$

$$\Delta(X) = \max_{x \in X} \sum_{r \in \mathcal{R}} (A_{\max}(r, x)/d_{\max}(r, x))^2 \quad \left(\begin{array}{c}\text{sum over } \mathcal{R} \text{ of squared} \\ \text{max (total / single) alloc}\end{array}\right)$$

Applicability in Practice: Computing β and γ is hard

Option 1: Overestimating β and γ

→ bad solution returned after few iterations

Option 2: Underestimating β and γ

→ no solution returned after *many* iterations

Approximation Algorithm for VNEP

Randomized Rounding Approximation

Algorithm: VNEP Approximation

```
// perform preprocessing
```

compute *optimal* LP solution

compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution

do

  solution ← RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)

while (solution *not* (α, β, γ)-approximate and rounding tries not exceeded)

Definition of Parameters

$\alpha = 1/3$ (relative achieved profit)

$\beta = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(R_S^V) \cdot \log(|R_S^V|)})$ (max node load)

$\gamma = (1 + \varepsilon \cdot \sqrt{2 \cdot \Delta(E_S) \cdot \log(|E_S|)})$ (max edge load)

$\varepsilon = \max_{r \in \mathcal{R}, x \in R_S} d_{\max}(r, x)/c_S(x) \leq 1$ (max demand/capacity)

$\Delta(X) = \max_{x \in X} \sum_{r \in \mathcal{R}} (A_{\max}(r, x)/d_{\max}(r, x))^2$ (sum over $\mathcal{R}$ of squared max (total / single) alloc)

Applicability in Practice: Computing β and γ is hard

Option 1: Overestimating β and γ

→ bad solution returned after few iterations

Option 2: Underestimating β and γ

→ no solution returned after *many* iterations

Option 3: Consider Heuristics

Return best solution found within X iterations.

Derived Heuristics

Randomized Rounding Approximation

Algorithm: VNEP Approximation

```
// perform preprocessing
```

compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution
do
  solution $\leftarrow$ RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)
while (solution *not* (α, β, γ)-approximate and rounding tries not exceeded)

Derived Heuristics

Heuristic Idea: Return best of X

Algorithm: Heuristic Adaptation

```
// perform preprocessing
```
compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution
do
  solution $\leftarrow$ RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)
while *rounding tries not exceeded*
return *best* solution

Vanilla Rounding: $\mathrm{RR}_{\mathrm{MinLoad}}$

- still may exceed capacities
- return solution with least resource violations (among those: highest profit)

Derived Heuristics

Heuristic Idea: Return best of X

Algorithm: Heuristic Adaptation

```
// perform preprocessing
```

compute *optimal* LP solution
compute $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ from LP solution
do
  solution $\leftarrow$ RoundingProcedure($\{\mathcal{D}_r\}_{r \in \mathcal{R}}$)
while *rounding tries not exceeded*
return *best* solution

Algorithm: RoundingProcedure (Heuristic)

Input : Optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$
foreach $r \in \mathcal{R}$ **do**
  choose m_r^k **with probability** f_r^k
  discard mapping if capacity violated
end
return *solution*

Vanilla Rounding: $\mathrm{RR}_{\mathrm{MinLoad}}$

- still may exceed capacities
- return solution with least resource violations (among those: highest profit)

Heuristic Rounding: $\mathrm{RR}_{\mathrm{Heuristic}}$

- $\mathrm{RoundingProcedure}$: discard chosen mappings exceeding capacities
- always yields feasible solutions
- return solution with highest profit

Taking a Step Back: How to compute LP Solutions?

Taking a Step Back: How to Compute LP Solutions?

Assumption (for now):

Sets of valid mappings are of polynomial size and given.
$\Rightarrow$ LP Formulation can be solved in polynomial-time.

How to compute optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$?

Taking a Step Back: How to Compute LP Solutions?

How to compute optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$?

Obtaining convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ is challenging!

1. Presented LP has exponential size and cannot be used.
2. Classic LP formulation may yield **meaningless** solutions for **cyclic** graphs:
   - **Theorem**: Solution to classic LP Formulation **cannot be decomposed** into valid mappings.
   - **Theorem**: Classic LP Formulation has **infinite integrality gap**.

Taking a Step Back: How to Compute LP Solutions?

How to compute optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$?

Obtaining convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$ is challenging!

1. Presented LP has exponential size and cannot be used.
2. Classic LP formulation may yield **meaningless** solutions for **cyclic** graphs:
   - **Theorem**: Solution to classic LP Formulation **cannot be decomposed** into valid mappings.
   - **Theorem**: Classic LP Formulation has **infinite integrality gap**.

Classic LP Formulation

Formulation 1: Classic MCF Formulation for the VNEP

$$\max \sum_{r \in \mathcal{R}} p_r x_r \qquad (5)$$

$$\sum_{u \in V_S^i} y_{r,i}^u = x_r \qquad \forall r \in \mathcal{R}, i \in V_r \qquad (6)$$

$$\sum_{u \in V_S \setminus V_S^i} y_{r,i}^u = 0 \qquad \forall r \in \mathcal{R}, i \in V_r \qquad (7)$$

$$\begin{bmatrix} \sum_{(u,v) \in \delta^+(u)} z_{r,i,j}^{u,v} \\ - \sum_{(v,u) \in \delta^-(u)} z_{r,i,j}^{v,u} \end{bmatrix} = \begin{bmatrix} y_{r,i}^u \\ -y_{r,j}^u \end{bmatrix} \forall \begin{bmatrix} r \in \mathcal{R}, (i,j) \in E_r, \\ u \in V_S \end{bmatrix} \qquad (8)$$

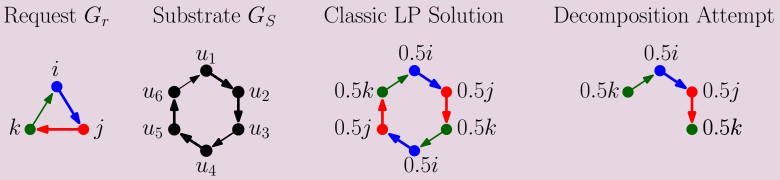
$$z_{r,i,j}^{u,v} = 0 \qquad \forall \begin{bmatrix} r \in \mathcal{R}, (i,j) \in E_r, \\ (u,v) \in E_S \setminus E_S^{i,j} \end{bmatrix} \qquad (9)$$

$$\sum_{i \in V_r, \tau_r(i) = \tau} c_r(i) \cdot y_{r,i}^u = a_r^{\tau,u} \qquad \forall r \in \mathcal{R}, (\tau, u) \in R_S^V \qquad (10)$$

$$\sum_{(i,j) \in E_r} c_r(i,j) \cdot z_{r,i,j}^{u,v} = a_r^{u,v} \qquad \forall r \in \mathcal{R}, (u,v) \in E_S \qquad (11)$$

$$\sum_{r \in \mathcal{R}} a_r^{x,y} \leq c_S(x,y) \; \forall (x,y) \in R_S \qquad (12)$$

Structural Deficiency of Classic LP Formulation



Taking a Step Back: How to Compute LP Solutions?

How to compute optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$?

Novel Decomposable Linear Programming Formulation (Details in the paper)

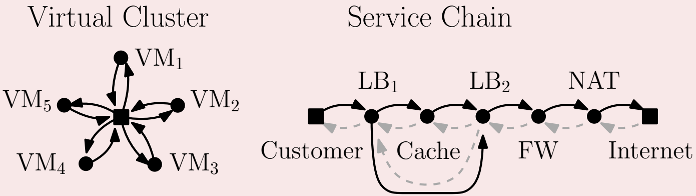
- **Intuition – 'breaking cycles'**: fix any node on a cycle $\rightarrow |V_S|$ copies of the classic Formulation.
- Formulation size increases by factor $\mathcal{O}(|V_S|)$ and **is only applicable for cactus request graphs**

Taking a Step Back: How to Compute LP Solutions?

How to compute optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$?

Novel Decomposable Linear Programming Formulation (Details in the paper)

- **Intuition – ‘breaking cycles’**: fix any node on a cycle $\rightarrow |V_S|$ copies of the classic Formulation.
- Formulation size increases by factor $\mathcal{O}(|V_S|)$ and **is only applicable for cactus request graphs**



cactus graphs: cycles intersect in at most one node

Taking a Step Back: How to Compute LP Solutions?

How to compute optimal convex combinations $\{\mathcal{D}_r\}_{r \in \mathcal{R}}$?

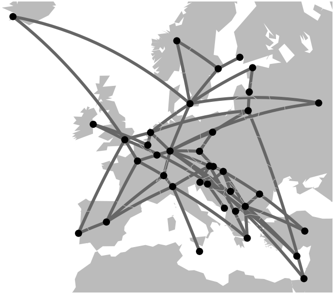
Novel Decomposable Linear Programming Formulation (Details in the paper)

- **Intuition – 'breaking cycles'**: fix any node on a cycle $\rightarrow |V_S|$ copies of the classic Formulation.
- Formulation size increases by factor $\mathcal{O}(|V_S|)$ and **is only applicable for cactus request graphs**
- Generalization to arbitrary request graphs is possible[a], but ...
  - Formulation size increases **super-polynomially** $\rightarrow$ **fixed-parameter tractable** approximations.
  - No polynomial-time approximations can exist **for arbitrary request graphs**, unless $\mathcal{P} = \mathcal{NP}$.

[a] Matthias Rost and Stefan Schmid. *(FPT-)Approximation Algorithms for the Virtual Network Embedding Problem*. Tech. rep. Mar. 2018. url: http://arxiv.org/abs/1803.04452.

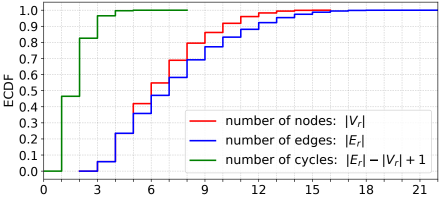
Computational Evaluation

Substrate: GEANT Network



Code available at
`https://github.com/vnep-approx/evaluation-ifip-networking-2018`

Requests: Synthetic Cactus Requests[3]



Generation Parameters for 1,500 instances

Number of requests: 40, 60, 80, 100

Node-Resource Factor (NRF): 0.2, 0.4, 0.6, 0.8, 1.0

Edge-Resource Factor (ERF): 0.25, 0.5, 1.0, 2.0, 4.0

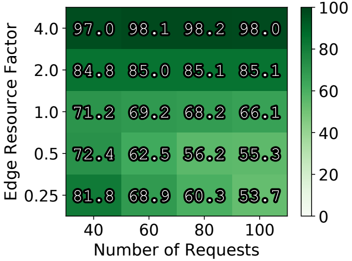
Instances per combination: 15

[3] Matthias Rost and Stefan Schmid. *Virtual Network Embedding Approximations: Leveraging Randomized Rounding*. Tech. rep. Mar. 2018. url: http://arxiv.org/abs/1803.03622

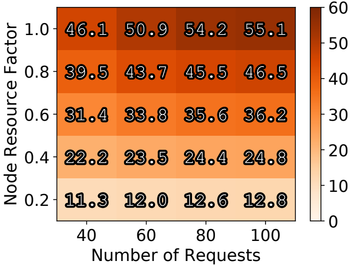
Computational Evaluation

Baseline Algorithm – $\mathrm{MIP}_{\mathrm{MCF}}$: solve classic MIP Formulation for upto 3 hours

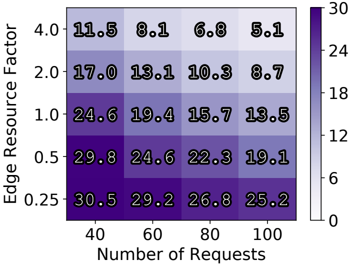
Acceptance Ratio



Avg. Node Load[3]

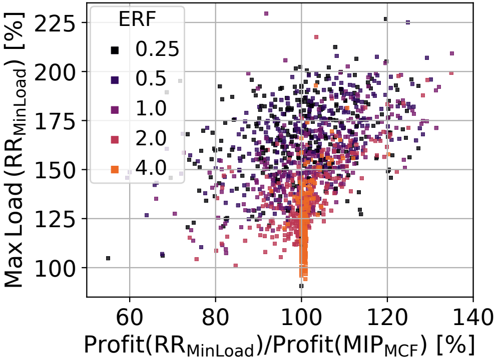


Avg. Edge Load[3]



[3] Matthias Rost and Stefan Schmid. *Virtual Network Embedding Approximations: Leveraging Randomized Rounding*. Tech. rep. Mar. 2018. url: http://arxiv.org/abs/1803.03622

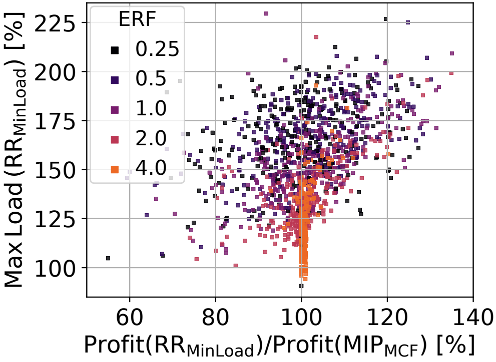
Vanilla Rounding Performance



- Relative profit ≈ 80 - 120%
- Resource augmentations mostly < 200%

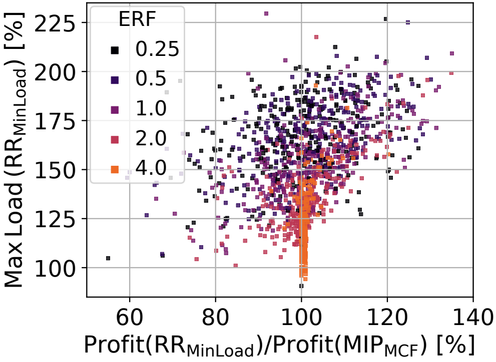
Computational Evaluation: Results

Vanilla Rounding Performance



- Relative profit ≈ 80 - 120%
- Resource augmentations mostly < 200%

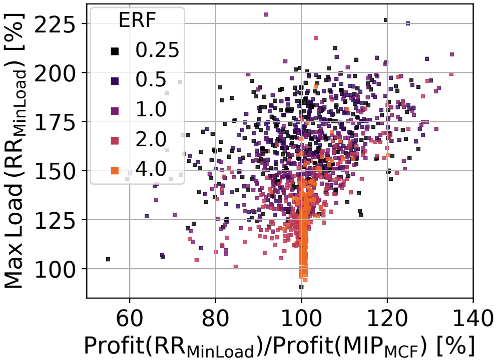
Vanilla Rounding Performance



- Relative profit $\approx$ 80 - 120%
- Resource augmentations mostly $<$ 200%

Computational Evaluation: Results

Vanilla Rounding Performance



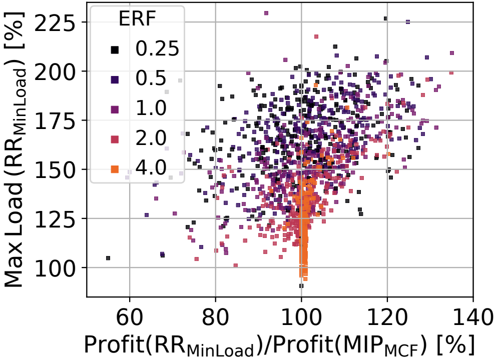
- Relative profit ≈ 80 - 120%
- Resource augmentations mostly < 200%

Heuristic Rounding (w/o augmentations)

- Relative profit ≈ 65 - 90%
- min: 22.5% / mean: 73.8% / max: 101%

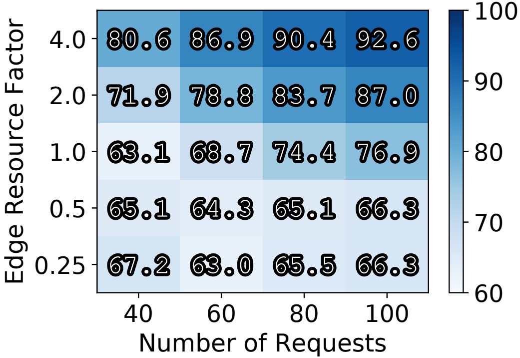
Computational Evaluation: Results

Vanilla Rounding Performance



- Relative profit ≈ 80 - 120%
- Resource augmentations mostly < 200%

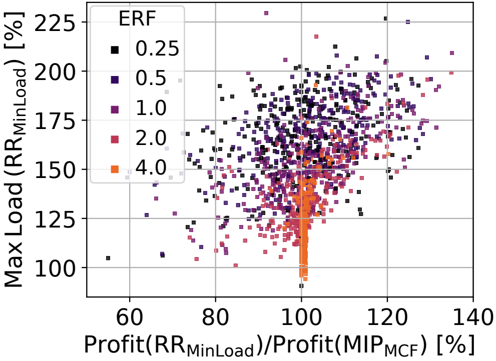
Heuristic Rounding (w/o augmentations)



- Relative profit ≈ 65 - 90%
- min: 22.5% / mean: 73.8% / max: 101%

Computational Evaluation: Results

Vanilla Rounding Performance



- Relative profit ≈ 80 - 120%
- Resource augmentations mostly $< 200\%$

Heuristic Rounding (w/o augmentations)

- Relative profit ≈ 65 - 90%
- min: 22.5% / mean: 73.8% / max: 101%

Conclusion

Conclusion: A First Step Towards *provably* Good Algorithms for the VNEP!

Contributions of our paper

1. **First approximation algorithm for the offline VNEP for maximizing the profit.**
2. **Derived heuristics (w/o) resource augmentations achieves 73.8% on average.**

Main Challenge: Computing Decomposable LP Solutions

Classic LP Formulation

- non-decomposable solutions
- infinite integrality gap

Novel LP Formulation

- decomposable formulation for cactus request graphs
- formulation size increases by factor $\mathcal{O}(|V_S|)$
- generalization to arbitrary request graphs possible[4]

Future Work

Other Rounding Heuristics / Column Generation for Solving the LP / Online Problem

[4] Matthias Rost and Stefan Schmid. *(FPT-)Approximation Algorithms for the Virtual Network Embedding Problem*. Tech. rep. Mar. 2018. url: http://arxiv.org/abs/1803.04452

Conclusion: A First Step Towards *provably* Good Algorithms for the VNEP!

Contributions of our paper

1. **First approximation algorithm for the offline VNEP for maximizing the profit.**
2. **Derived heuristics (w/o) resource augmentations achieves 73.8% on average.**

Main Challenge: Computing Decomposable LP Solutions

Classic LP Formulation

- non-decomposable solutions
- infinite integrality gap

Novel LP Formulation

- decomposable formulation for cactus request graphs
- formulation size increases by factor $\mathcal{O}(|V_S|)$
- generalization to arbitrary request graphs possible[4]

Future Work

Other Rounding Heuristics / Column Generation for Solving the LP / Online Problem

[4] Matthias Rost and Stefan Schmid. *(FPT-)Approximation Algorithms for the Virtual Network Embedding Problem*. Tech. rep. Mar. 2018. url: http://arxiv.org/abs/1803.04452

Conclusion: A First Step Towards *provably* Good Algorithms for the VNEP!

Contributions of our paper

1. **First approximation algorithm for the offline VNEP for maximizing the profit.**
2. **Derived heuristics (w/o) resource augmentations achieves 73.8% on average.**

Main Challenge: Computing Decomposable LP Solutions

Classic LP Formulation

- non-decomposable solutions
- infinite integrality gap

Novel LP Formulation

- decomposable formulation for cactus request graphs
- formulation size increases by factor $\mathcal{O}(|V_S|)$
- generalization to arbitrary request graphs possible[4]

Future Work

Other Rounding Heuristics / Column Generation for Solving the LP / Online Problem

[4] Matthias Rost and Stefan Schmid. *(FPT-)Approximation Algorithms for the Virtual Network Embedding Problem*. Tech. rep. Mar. 2018. url: http://arxiv.org/abs/1803.04452

References I

- Raghavan, P and C D Thompson. "Provably Good Routing in Graphs: Regular Arrays". In: *Proc. 17th ACM STOC*. 1985, pp. 79–87.
- Rost, Matthias and Stefan Schmid. "Charting the Complexity Landscape of Virtual Network Embeddings". In: *Proc. IFIP Networking*. 2018.
- – .*(FPT-)Approximation Algorithms for the Virtual Network Embedding Problem*. Tech. rep. Mar. 2018. url: http://arxiv.org/abs/1803.04452.
- – .*Virtual Network Embedding Approximations: Leveraging Randomized Rounding*. Tech. rep. Mar. 2018. url: http://arxiv.org/abs/1803.03622.